Cloud Basic Project
Implementing a Cloud-Based File Storage System

Alessandro Della Siega

March 2024

1 Introduction

This project is about implementing a cloud-based file storage system. Our goal is to enable file upload, download, and deletion while ensuring user privacy with individual storage spaces and emphasizing scalability, security, and cost-efficiency. We'll take into consideration the Nextcloud platform. Through deployment, testing, and analysis, we aim to achieve coherent integration and the desired performance.

2 Software

2.1 Nextloud

Nextcloud is well-suited for deployment within containerized environments. It offers Docker images and documentation to facilitate deployment using containerization services such as Docker. Containers are an optimal tool to run this service because they are portable and self-sufficient units that package the application and its dependencies, allowing it to run consistently across different environments.

2.2 Locust

To test and analyze the behaviour of the deployed infrastructure, we use Locust. Locust is an open-source tool for load testing web applications and APIs. It allows users to simulate multiple concurrent users accessing a web application to assess its scalability and performance. With its Python-based scripting and distributed architecture, Locust provides real-time insights into performance metrics. We just need to write a python script in which we define the tasks assigned to the test users.

2.3 Docker Compose file

The crucial piece of software is the Docker Compose file. It includes two services: Nextcloud and Locust. Nextcloud service uses the latest Nextcloud image and setting environment variables for the admin user and SQLite database. Locust service uses the Locust image and runs the Locust task file to simulate load testing on the Nextcloud service. It mounts volumes for Locust task scripts and test data. Both services are defined within the services section and utilize volumes for data persistence.

3 Manage User and Files

Nextcloud offers the following operations by default:

- users can sign up, log in, and log out;
- users can be organized into groups, simplifying access control and collaboration, Nextcloud allows administrators to create manage, and delete users within the platform;
- users are able to upload, download and delete files from their private storage;

4 Address Scalability

4.1 Handling a growing number of users and files

To handle a growing number of users and files we can consider many solution.

Horizontal Scaling: invest resources in order to obtain multiple nodes on which distribute the growing amount of workload. This approach could guarantee a fault-tolerant system but requires tools to orchestrate the multiple node architecture. This approach is optimal over time.

Vertical Scaling: invest resources in CPU, RAM, and disk to obtain more computational power and storage space. We have an immediate solution but we still have a limit in our computing resources given by the hardware that we own. This approach is optimal if we need a low-cost solution in a short time.

Database Scalability: implement a scalable database solution such as MySQL or PostgreSQL instead of SQLite (to be considered just a temporary solution in order to test the service) to handle growing data volumes efficiently.

4.2 Handling of Increased Load and Traffic

Load Balancing: if we adopt a horizantal scaling approach, we could implement a load balancer in front of Nextcloud instances to distribute incoming requests evenly across multiple servers. This ensures that no single server becomes a bottleneck, and the system can handle a larger number of concurrent users.

Caching: implement caching mechanisms at various layers of the application stack to reduce the load on backend servers and improve response times. Utilize caching solutions like Redis to cache frequently accessed data and static content, reducing the number of requests that need to be processed by the backend servers.

5 Address Security

Nextcloud offers the following solution to tackle security issues.

Server-Side Encryption: we cann implement a secure file storage and transmission system by enabling server-side encryption in Nextcloud. This ensures that even if the server is compromised, the data remains protected.

Two-Factor Authentication: implement 2FA in Nextcloud to enhance user authentication security. Require users to provide a second form of verification, such as a one-time password generated by a mobile authenticator app, in addition to their password.

Password Policy Enforcement: define strong password policies in Nextcloud to ensure that users create and maintain secure passwords. This includes requirements such as minimum length, complexity, and expiration periods.

OAuth 2.0 Clients: implement OAuth 2.0 authentication for Nextcloud to allow users to authenticate using third-party identity providers. This reduces the risk of password-related vulnerabilities and enhances user convenience.

Analysis of Activities and Logins: a measure to prevent unauthorized access is to use tools to analyze activities and logins performed by users. In case of suspicious activities or accesses in the system we can take the necessary countermeasures.

6 Cost-Efficiency

The cost implications of our design are very small, we just need one node (a computer) accessible from internet. We are assuming that the number of the users is small and the activities performed by them are not computationally expensive. Otherwise, we have to face the costs of scaling our system as discussed before.

Some approaches to optimise the system for cost efficiency are

Resource Usage Optimization: optimize resource utilization by rightsizing instances and containers based on workload requirements. Utilize monitoring tools to identify underutilized resources and scale them down accordingly to minimize costs.

Pay-As-You-Go Model: leverage cloud services that offer pay-as-you-go pricing models, allowing you to pay only for the resources you consume. For instance we could use a Object Storage service on the cloud to pay for just what we use. In this case we would also reduce the workload on our node.

Continuous Monitoring: use monitoring tools like Prometheus, Grafana, or cloud provider-specific monitoring services to gain insights into cost drivers and identify opportunities for optimization.

7 Deployment

The steps that define a basic deployment plan are:

1. install Docker and Docker Compose on the system;
2. write the `docker-compose.yml` file which includes the Nextcloud service;
3. execute `docker-compose -up` ;
4. the service is active and it can be accessed at `http://localhost:8080`.

Now, we have to monitor and manage the deployed system. The most important information that we need monitor are the users, the actions that each user takes and the amount of storage required. Nextcolud offers by default such analysis. Obviously, every user will need a different amount of storage. If the amount of workload requires a lot of resources, we can tackle this problem by offering a pay-as-you-go to the users in order to pay our providers.

Given a low budget and basic existing infrastructure, a cost-effective cloud provider with straightforward pricing and minimal management overhead would be ideal. Among the many solutions like Amazon Web Services, Azure, Google Cloud Platform, we pick AWS. Amazon Web Services (AWS) is a leading cloud platform that offers a wide range of services and features, making it a compelling choice for deploying a Nextcloud-based file storage system. Here are several reasons to justify the use of AWS for this purpose:

- AWS provides a vast array of services that can support the deployment of Nextcloud, including compute (EC2), storage (S3), databases (RDS), networking (VPC), and security (IAM)
- AAWS operates a global network of data centers across multiple regions and availability zones, enabling to deploy Nextcloud instances closer to users for reduced latency and improved performance.
- AWS's infrastructure is designed for scalability, allowing to scale our Nextcloud deployment up or down based on demand. With services like Auto Scaling and AWS Lambda, it is possible to automate resource provisioning and optimize costs by only paying for the resources that are used
- AWS prioritizes security, providing a wide range of security features and certitications. These include network isolation with Virtual Private Cloud (VPC), data encryption at rest and in transit, identity management with IAM, and compliance certifications such as SOC, ISO, and PCI DSS

8 Test of the Infrastructure

By defining the Locust task script, we can set the operations that our test users will perform. We create 3 files of different size (1kB, 1MB and 1GB) and we set 1 locust worker, 30 users and we try to make different actions using such files ramping up 1 user each second. The results are the following.

Table 1: Summary of Request Statistics for 1GB file

Type	**Requests**	**Fails**	**Median(ms)**	**Average(ms)**	**Min(ms)**	**Max(ms)**
PROPFIND	68	16	11000	14238.75	4531	31000
PUT	101	71	27000	27116.26	3849	39692

Table 2: Summary of Request Statistics for 1MB file

Type	**Requests**	**Fails**	**Median(ms)**	**Average(ms)**	**Min(ms)**	**Max(ms)**
PROPFIND	2827	104	64	730.45	16	27955
DELETE	474	9	260	487.53	16	3588
GET	2270	2	53	289.92	17	1048576
PUT	333	3	280	2379.96	7	27000

Table 3: Summary of Request Statistics for 1kB file

Type	**Requests**	**Fails**	**Median(ms)**	**Average(ms)**	**Min(ms)**	**Max(ms)**
PROPFIND	250	1	31	122.88	21	1479
DELETE	244	23	250	420.44	15	2326
GET	1200	33	54	119.15	16	1982
PUT	286	31	250	405.47	31	3931

Overall, it's evident that performance deteriorates for operations involving larger files. This degradation in performance can be attributed to the increased processing and transmission time required for the CPU to handle larger file sizes, resulting in huge response times and failure rates. Optimal strategies to mitigate these issues may involve optimizing server configurations, implementing efficient file handling algorithms, and possibly introducing mechanisms to manage large file transfers more effectively.